

REMARKS

Applicants respectfully request reconsideration of the above referenced patent application in view of the amendments and remarks set forth herein, and respectfully request that the Examiner withdraw all rejections. No claims have been amended. No claims have been canceled. No claims have been added. Thus, claims 1-30 are pending.

35 U.S.C. §103(a) Rejections

35 U.S.C. §103(a) Rejection over *Starr* in view of *Hass*

The Office Action rejects claims 1-30 under 35 U.S.C. §103(a) as allegedly being obvious in view of *Starr* et al., USPN 2005/0122986 (hereinafter "*Starr*") in view of *Hass*, USPN 2005/0027793 (hereinafter "*Hass*"). As described in greater detail below, *Starr* does not qualify as prior art. Therefore, the rejection of claims 1-30 as allegedly being obvious in view of *Starr* and *Hass* is moot.

Pursuant to 37 C.F.R. §1.131, Applicants submit as Exhibit A in the Appendix below pages 4-9 of a confidential internal document of the Assignee entitled "Hardware Multi-Threading for Packet Processing." Applicants also submit as Exhibit B in the Appendix below respective declarations from each of Applicants Yatin Hoskote, Sriram R. Vangal, Vasantha K. Erraguntla and Nitin Y. Borkar to establish Applicants' conception of the claimed invention before the December 5, 2003 filing date of *Starr*.

Applicants respectfully submit that the exhibits found in the Appendix below show that Applicants conceived of the claimed invention prior to the December 5, 2003 filing date of *Starr*, and that *Starr* therefore does not qualify as a prior art reference. In view of the foregoing, Applicants request that the rejection of claims 1-30 as allegedly being anticipated by *Starr* and *Hass* be withdrawn.

CONCLUSION

For at least the foregoing reasons, Applicants submit that the objections and rejections have been overcome. Therefore, claims 1-30 are in condition for allowance and such action is earnestly solicited. The Examiner is respectfully requested to contact the undersigned by telephone if such contact would further the examination of the present application. Please charge any shortages and credit any overcharges to our Deposit Account number 02-2666.

Respectfully submitted,
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP

Date: July 8, 2008

/Dermot G. Miller/
Dermot G. Miller
Attorney for Applicants
Reg. No. 58,309

1279 Oakmead Parkway
Sunnyvale, CA 94085-4040
(503) 439-8778

Appendix:

Please see Exhibit A and Exhibit B, included hereafter.

EXHIBIT "A"

Hardware Multi-Threading for Packet Processing

Inventor(s): Yatin Hoskote, Sriram Vangal, Vasantha Erraguntla and Nitin Borkar

A. General Purpose(s) of Invention(s)

A unique hardware multi-threading approach to network protocol processing is presented in this invention disclosure that implements the transmission control protocol (TCP) input and output processing using a programmable hardware engine. Protocol processing is a complex task that strains both compute resources and memory bandwidth, especially with network bandwidth increasing exponentially. The approach described here:

1. Uses multiple threads to switch between Ethernet packets and thus better utilize the execution core by hiding latency from memory accesses and other high latency functions.
2. Implements the thread suspension, scheduling and save/restore in hardware so that the programmer is not burdened with the responsibility.
3. Ensures that all computing resources are completely dedicated to the thread currently being processed rather than being shared between all existing threads.
4. Enables the packet processing engine to process packet receives and transmits at 10+Gbps Ethernet traffic rate at a client or server end.
5. Minimizes the need for buffering by enabling the engine to provide line-speed TCP/IP processing for all packets above 512 bytes.
6. Enables multiple engines to be used as part of a multi-core solution to scale up to higher Ethernet bandwidth and smaller packet sizes.
7. Is a key enabler for a low power/high performance solution with better MIPS/Watt than a general purpose CPU.
8. Can be easily extended to handle emerging protocols including iSCSI and RDMA.

B. Advantages(s) of the Invention(s) over prior art

Transmission Control Protocol (TCP) is a connection-oriented reliable protocol accounting for over 80% of the network traffic. Today TCP processing is performed almost exclusively through software. Even with the advent of GHz processor speeds there is a need for dedicated TCP offload engine (TOE) to support high bandwidths (of 10Gb/s and beyond). Memory performance is also not keeping up with processors. Making the TOE programmable improves its flexibility and also simplifies its design. At 10Gb/s, a minimum-size Ethernet packet arrives every 67.2ns. Processing each packet requires the engine to access data from host memory in addition to the execution of several hundred instructions. Each host memory access itself can take up to 100ns. In order to keep up with the packet rate, a mechanism to hide the memory latency is essential. Such a mechanism can also be used to hide latency from other functions such as data transfers, timer events and so on. We propose a novel multi-threaded TOE architecture for the engine to implement the complete TCP input and output processing. This approach expedites inbound and outbound packet processing, minimizing the need for costly buffering and queuing. Unlike conventional approaches to multi-threading, **this approach implements the multiple thread mechanism entirely in hardware, including thread suspension, scheduling and save/restore of thread state.** This frees the programmer from the responsibility of maintaining threads, as well as removes the element of human error. The programming model is thus far simpler than the more common model of a programmer or compiler generating multi-threaded code. The same code that runs on a single-threaded engine will run on this engine, but with much greater efficiency. The overhead penalty from switching between threads is also minimized in this approach resulting in better throughput.

C. Essential Element(s) or Key(s) to the Invention

We first look at a system view of the TOE based solution. Figure 1 shows a computer system consisting of a processor with a memory unit, a chipset and a network interface card (NIC). The TOE can be physically part of the processor, the chipset or the NIC. Of the three possible options, the TOE as part of the chipset would provide better access to host memory in the configuration shown. We expect the TOE to be eventually absorbed into the CPU in future. This system receives and transmits Ethernet data via the NIC interface. Upon

completion of MAC level and significant IP layer processing by the NIC, the packet is forwarded for TCP layer (and above) processing to the TOE.

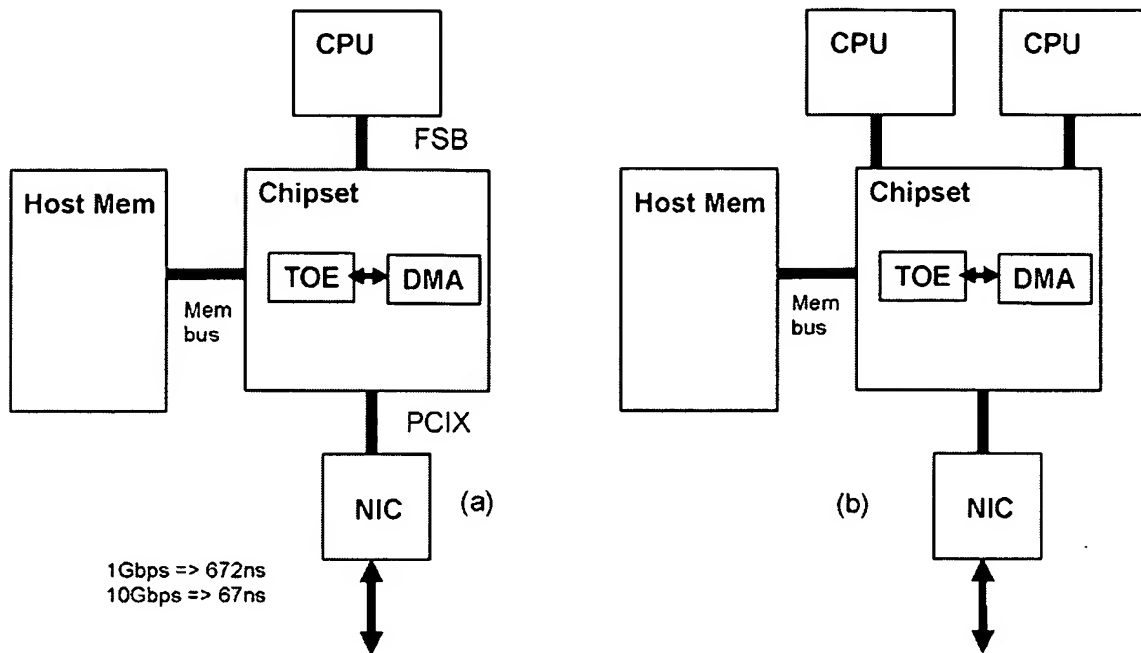


Figure 1. System level view of a TOE with DMA capability in (a) Single CPU (b) Dual CPU system.

Top-level block diagram showing architecture of the TOE is shown in Figure 2. To minimize buffering needs, the design uses a dual frequency design with an execution core running at a higher speed than the peripheral units. The execution core performs the heart of TCP processing under directions of the instruction stored in its instruction cache. The design also uses a large on-die cache (L3) to store TCP connection context, which provides temporal locality for a few thousand connections, with L3 size limited only by allowable physical area. The context is portion of the Transmission control block (TCB) that TCP is required to maintain for each connection. Caching this context on-die is critical for 10+Gb/s performance. The scheduler is the central control unit -- a "traffic cop" that directs packets between different blocks in the design. The processing pipeline diagrams in Figure 3 show the different steps in processing incoming and outgoing packets.

1. On receives, the NIC queues incoming TCP/IP header and payload data in the inbound queue. Header information is forwarded to the scheduling unit.
2. A hash based lookup is performed using the header descriptors to correlate the packet with its connection. The scheduler performs a lookup on the L3 cache. On a miss, a lookup against hashed entries in host memory is scheduled via the memory queue. In either case, the fetched context is loaded into a wide (~512B) working register in the core.
3. The execution core also programs the DMA control unit. Payload data is then transferred from the internal receive buffers to locations in host memory using DMA. It is important to note that TCP processing happens in parallel with the DMA operation, as shown in Figure 3(a). This concurrent DMA transfer is critical for high performance.
4. The architecture presents three queues as a hardware mechanism to interface with the TOE driver and operating system (OS). A doorbell queue (DBQ) to accept send (transmit) requests, and a completion queue (CQ) and an exception/event queue (EQ). Upon completion of TCP processing, the scheduler updates CQ with the completion descriptors and the EQ with the status of completion (pass/fail), which can generate a processor interrupt. Events can either be exception status or Interrupts. The driver may choose to coalesce the events and interrupts for more efficient processing. At this point, the context is also updated with the processing results and written back to the L3 cache, completing input processing.

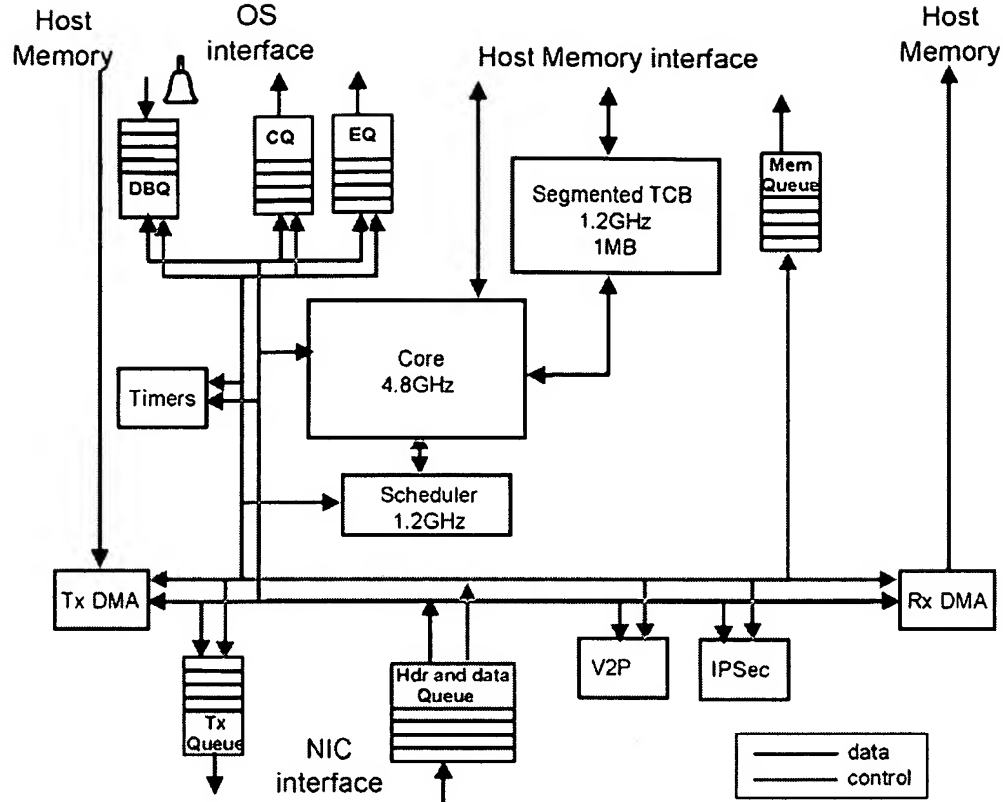


Figure 2. Top level block diagram of the TCP offload engine.

Host initiated packet sends consist of the following steps

1. The OS uses the TOE driver to place doorbell descriptors in the doorbell queue (DBQ). The doorbell contains pointers to either the Tx or Rx descriptors queues (depending on send or receive DB), which reside in host memory. The TOE is responsible for fetching and caching the descriptors in L3. As shown in processing pipeline diagram in Figure 3 (b), we next identify the connection by scheduling a lookup against the local L3 cache. The connection context is loaded into the core working register, starting core processing.
2. The execution engine then performs the heart of TCP output processing under programmed control at high speed. Response ACK headers are generated by the core. The core interacts with the timer block to set/clear timers. The core also programs the DMA control unit by building the appropriate descriptor ring and queues the Tx DMA requests. Here again, for low latency, payload data is transferred from the payload locations in host memory to internal transmit buffers using DMA. It is also important to note that TCP processing happens in parallel with the DMA operation, as in Figure 3(b).
3. Completion notification of send is accomplished by populating CQ and EQs to signal end of transmit.

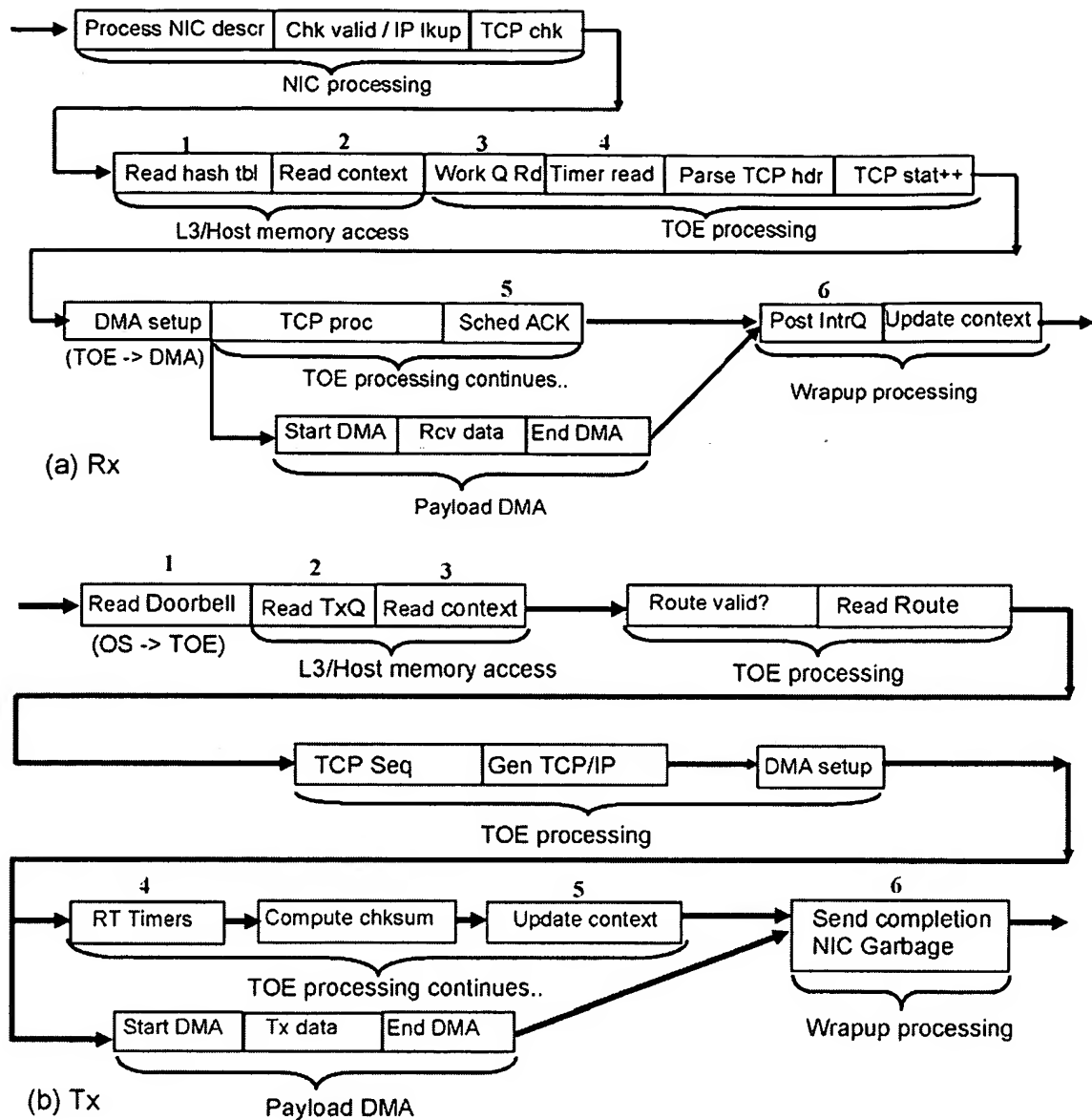


Figure 3. Packet processing pipeline on (a) packet receive and on (b) packet send.

As can be seen in the pipeline diagrams in Figure 3, there are several memory accesses as well as synchronization points with the DMA engine or timers that can cause the execution core to stall while waiting for a response. In an effort to hide latencies from these memory accesses and other tasks and dramatically improve throughput, multiple threads are utilized. Thread switches on both Tx and Rx can happen when waiting on outstanding memory requests or on pending DMA transactions or on completion events from other hardware assist blocks such as timers. Six of these points are highlighted and numbered in both the transmit and receive pipelines in Figure 3. If core processing completes prior to DMA (Figure 3), thread switch can occur to improve throughput. When DMA ends, the thread switches back to update the context with processed results and the

context is written back to the L3 cache. Unlike conventional approaches, the scheduler controls the switching between different threads.

Scheduling Logic:

A thread is associated with each network packet that is being processed, both incoming and outgoing. This differs from other approaches that associate threads with each task to be performed, irrespective of the packet. The scheduler spawns a thread when a packet belonging to a new connection needs to be processed. A second packet for that same connection will not be assigned a thread until the first packet is completely processed and the updated context has been written back to L3 cache. This is under the control of the scheduler. When the processing of a packet in the core is stalled, the thread state is stored to temporary memory and the scheduler will spawn a thread for a packet on a different connection. It could also wake up a thread for a previously suspended packet by restoring its state and allow it to run to completion. In this approach, the scheduler also spawns special maintenance threads for global tasks like gathering statistics on internet traffic. The priority mechanism to determine which packet to schedule next is programmed into the scheduler.

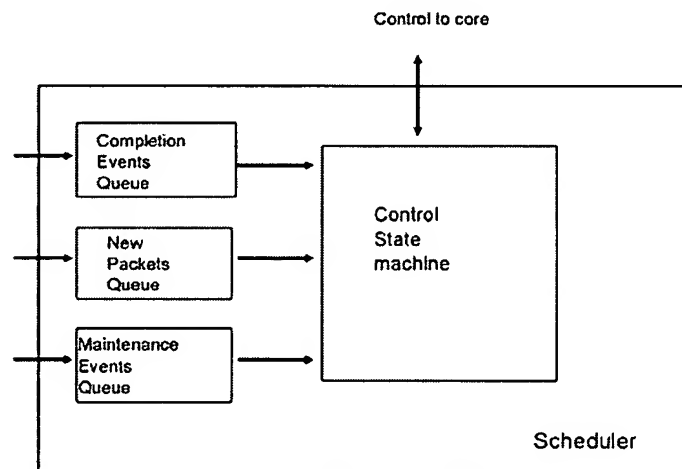


Figure 4. Scheduler

The scheduler has to arbitrate between events that wake up or spawn threads from the following categories (as shown in Figure 4):

1. New packets on new network connections or old connections with no active packets in the core
2. New packets on existing network connections with active packets in the core
3. Completion events for suspended threads
4. Maintenance events

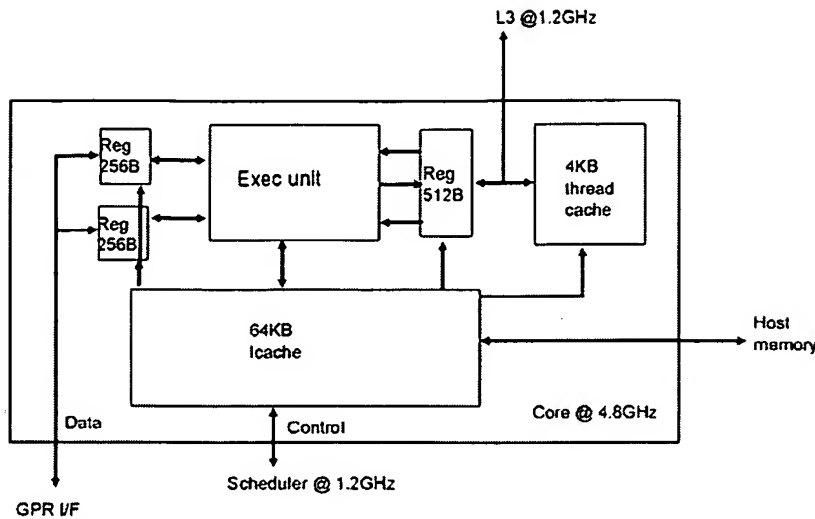


Figure 5. Core Engine

The execution core micro-architecture diagram is shown in Figure 5. The design provides a thread cache, running at core speed, which allows intermediate architecture state to be saved and restored for each thread. This cache is estimated to be 8-16 threads deep and 512byte wide. The width of the cache is determined by the amount of context information that needs to be saved for each packet. The depth of the cache is determined by the packet arrival rate. Analysis shows that for 256byte packets on a 10Gbps link for performing both receives and transmits, a 16 deep cache is sufficient because that is more than the number of packets that could be active at any point in time. In addition, the core consists of a working register, which contains the TCB context for the current active connection, a fully pipelined execution unit and significant number of scratch registers to save intermediate processed results of execution. The core is controlled by instructions issued by the control cache block. The design also provides a high-bandwidth connection between the thread cache and the working register making possible very fast and parallel transfer of thread state between the working register and the thread cache. This ensures that the overhead penalty from thread switches is minimal. At the sample frequencies shown here, the overhead penalty is less than 3ns. The working register, execution core and scratch registers are completely dedicated to the packet currently being processed. This is again different from other approaches where the resources are split up a priori and dedicated to specific threads. This ensures adequate resources for each packet without having to duplicate resources and increase engine die area.

D. Value of the Invention to Intel

This architecture enables network protocol processing for supporting 10+Gb/s Ethernet bandwidth and beyond at a client or server end. The design provides mechanisms to hide memory latency, in hardware and in a manner invisible to the programmer. The same code written for a single thread machine can run far more efficiently on this design without the need to rewrite it. The design and validation of the TOE is simpler in this approach than other approaches to multi-threading. The requirements on the compiler and the programming model are also far simpler. Multi-threading is critical to the ability of the offload engine to scale up to multi-gigabit Ethernet rates and enables implementation of a realistic TOE product, as part of Intel's long term TOE product roadmap. This solution is being implemented for use in CRL's Xtreme II prototype chip.

EXHIBIT "B"

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 10/814,496 Confirmation No. 6445
Applicant : Yatin Hoskote
Filed : March 31, 2004
TC/A.U. : 2616
Examiner : HARPER, Kevin C.

Docket No. : 42P18633
Customer No. :

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF VASANTHA K. ERRAGUNTLA
PURSUANT TO 37 C.F.R. §1.131

Sir:

I, Vasantha K. Erraguntla, hereby declare that:

1. Yatin Hoskote, Sriram R. Vangal, Nitin Y. Borkar and I are the co-inventors of the above-described patent application and the co-inventors of the subject matter described and claimed therein.
2. Intel Corporation, of Santa Clara, California, is the Assignee of the patent application described above.
3. I have been employed by Intel Corporation from prior to December 03, 2003.

4. At least prior to December 03, 2003, we conceived in this country (U.S.A.) the invention claimed in the above-described application.

5. As evidence of conception, attached hereto is Exhibit A. Exhibit A comprises pages 4-9 of a confidential internal document of the Assignee entitled "Hardware Multi-Threading for Packet Processing." These pages are representative of our inventive work and were created prior to December 03, 2003. Exhibit A describes a hardware multithreading approach to network protocol processing using a dedicated Transmission Control Protocol (TCP) offload engine (TOE). More particularly, the block diagram in Figure 2 on page 6, the packet processing diagram in Figure 3 of page 7, the scheduler diagram in Figure 4 of page 8 and the core engine diagram in Figure 5 of page 9 illustrate the invention as claimed in the above-described application. The description of Figures 2-5 on pages 5-9 further describes the invention as claimed in the above-described application.

For example, the figures and text disclose, *inter alia*, a dedicated hardware TOE system or apparatus configured to queue incoming TCP/IP header and payload data in an inbound queue. A hash lookup is performed to find any existing connection associated with inbound queued data. A fetched connection context may be retrieved based on lookups associated with the inbound queued data. As TCP processing takes place, dynamic memory access (DMA) operations also take place in parallel to exchange payload data with a host memory. These parallel operations provide at least some of the significant benefits over previous techniques. The figures and text of Exhibit A also discusses other more particular features variously related to, for example, various queues to exchange data with an interface between the TOE and an operating system (OS), a

scheduler of the TOE engine to variously spawn and/or wake up threads, and a core engine operating on the TOE.

6. From at least prior to December 03, 2003 to constructive reduction to practice (application filing on March 31, 2004), due diligence was taken in reducing the invention to practice.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-described application or an patent issued therefrom.

Respectfully submitted,

Date July 7th, 2008 
Vasantha K. Erraguntla

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 10/814,496 Confirmation No. 6445
Applicant : Yatin Hoskote
Filed : March 31, 2004
TC/A.U. : 2616
Examiner : HARPER, Kevin C.

Docket No. : 42P18633
Customer No. :

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF YATIN HOSKOTE
PURSUANT TO 37 C.F.R. §1.131

Sir:

I, Yatin Hoskote, hereby declare that:

1. Sriram R. Vangal, Vasantha K. Erraguntla, Nitin Y. Borkar and I are the co-inventors of the above-described patent application and the co-inventors of the subject matter described and claimed therein.

2. Intel Corporation, of Santa Clara, California, is the Assignee of the patent application described above.

3. I have been employed by Intel Corporation from prior to December 03, 2003.

4. At least prior to December 03, 2003, we conceived in this country (U.S.A.) the invention claimed in the above-described application.

5. As evidence of conception, attached hereto is Exhibit A. Exhibit A comprises pages 4-9 of a confidential internal document of the Assignee entitled "Hardware Multi-Threading for Packet Processing." These pages are representative of our inventive work and were created prior to December 03, 2003. Exhibit A describes a hardware multithreading approach to network protocol processing using a dedicated Transmission Control Protocol (TCP) offload engine (TOE). More particularly, the block diagram in Figure 2 on page 6, the packet processing diagram in Figure 3 of page 7, the scheduler diagram in Figure 4 of page 8 and the core engine diagram in Figure 5 of page 9 illustrate the invention as claimed in the above-described application. The description of Figures 2-5 on pages 5-9 further describes the invention as claimed in the above-described application.

For example, the figures and text disclose, *inter alia*, a dedicated hardware TOE system or apparatus configured to queue incoming TCP/IP header and payload data in an inbound queue. A hash lookup is performed to find any existing connection associated with inbound queued data. A fetched connection context may be retrieved based on lookups associated with the inbound queued data. As TCP processing takes place, dynamic memory access (DMA) operations also take place in parallel to exchange payload data with a host memory. These parallel operations provide at least some of the significant benefits over previous techniques. The figures and text of Exhibit A also discusses other more particular features variously related to, for example, various queues to exchange data with an interface between the TOE and an operating system (OS), a

scheduler of the TOE engine to variously spawn and/or wake up threads, and a core engine operating on the TOE.

6. From at least prior to December 03, 2003 to constructive reduction to practice (application filing on March 31, 2004), due diligence was taken in reducing the invention to practice.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-described application or an patent issued therefrom.

Respectfully submitted,

Date 7/7/, 2008 
Yatin Hoskote

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 10/814,496 Confirmation No. 6445
Applicant : Yatin Hoskote
Filed : March 31, 2004
TC/A.U. : 2616
Examiner : HARPER, Kevin C.

Docket No. : 42P18633
Customer No. :

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF NITIN Y. BORKAR
PURSUANT TO 37 C.F.R. §1.131

Sir:

I, Nitin Y. Borkar, hereby declare that:

1. Yatin Hoskote, Sriram R. Vangal, Vasantha K. Erraguntla, and I are the co-inventors of the above-described patent application and the co-inventors of the subject matter described and claimed therein.

2. Intel Corporation, of Santa Clara, California, is the Assignee of the patent application described above.

3. I have been employed by Intel Corporation from prior to December 03, 2003.

4. At least prior to December 03, 2003, we conceived in this country (U.S.A.) the invention claimed in the above-described application.

5. As evidence of conception, attached hereto is Exhibit A. Exhibit A comprises pages 4-9 of a confidential internal document of the Assignee entitled "Hardware Multi-Threading for Packet Processing." These pages are representative of our inventive work and were created prior to December 03, 2003. Exhibit A describes a hardware multithreading approach to network protocol processing using a dedicated Transmission Control Protocol (TCP) offload engine (TOE). More particularly, the block diagram in Figure 2 on page 6, the packet processing diagram in Figure 3 of page 7, the scheduler diagram in Figure 4 of page 8 and the core engine diagram in Figure 5 of page 9 illustrate the invention as claimed in the above-described application. The description of Figures 2-5 on pages 5-9 further describes the invention as claimed in the above-described application.

For example, the figures and text disclose, *inter alia*, a dedicated hardware TOE system or apparatus configured to queue incoming TCP/IP header and payload data in an inbound queue. A hash lookup is performed to find any existing connection associated with inbound queued data. A fetched connection context may be retrieved based on lookups associated with the inbound queued data. As TCP processing takes place, dynamic memory access (DMA) operations also take place in parallel to exchange payload data with a host memory. These parallel operations provide at least some of the significant benefits over previous techniques. The figures and text of Exhibit A also discusses other more particular features variously related to, for example, various queues to exchange data with an interface between the TOE and an operating system (OS), a

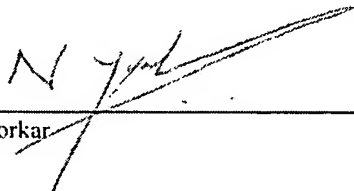
scheduler of the TOE engine to variously spawn and/or wake up threads, and a core engine operating on the TOE.

6. From at least prior to December 03, 2003 to constructive reduction to practice (application filing on March 31, 2004), due diligence was taken in reducing the invention to practice.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-described application or an patent issued therefrom.

Respectfully submitted,

Date July 7th, 2008, 2008



Nitin Y. Borkar

ATTN: Mr. Dermot Miller

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 10/814,496 Confirmation No. 6445
Applicant : Yatin Hoskote
Filed : March 31, 2004
TC/A.U. : 2616
Examiner : HARPER, Kevin C.

Docket No. : 42P18633
Customer No. :

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF SRIRAM R. VANGAL
PURSUANT TO 37 C.F.R. §1.131

Sir:

I, Sriram R. Vangal, hereby declare that:

1. Yatin Hoskote, Vasantha K. Erraguntla, Nitin Y. Borkar and I are the co-inventors of the above-described patent application and the co-inventors of the subject matter described and claimed therein.

2. Intel Corporation, of Santa Clara, California, is the Assignee of the patent application described above.

3. I have been employed by Intel Corporation from prior to December 03, 2003.

4. At least prior to December 03, 2003, we conceived in this country (U.S.A.) the invention claimed in the above-described application.

5. As evidence of conception, attached hereto is Exhibit A. Exhibit A comprises pages 4-9 of a confidential internal document of the Assignee entitled "Hardware Multi-Threading for Packet Processing." These pages are representative of our inventive work and were created prior to December 03, 2003. Exhibit A describes a hardware multithreading approach to network protocol processing using a dedicated Transmission Control Protocol (TCP) offload engine (TOE). More particularly, the block diagram in Figure 2 on page 6, the packet processing diagram in Figure 3 of page 7, the scheduler diagram in Figure 4 of page 8 and the core engine diagram in Figure 5 of page 9 illustrate the invention as claimed in the above-described application. The description of Figures 2-5 on pages 5-9 further describes the invention as claimed in the above-described application.

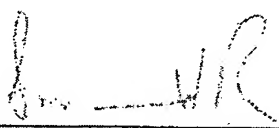
For example, the figures and text disclose, *inter alia*, a dedicated hardware TOE system or apparatus configured to queue incoming TCP/IP header and payload data in an inbound queue. A hash lookup is performed to find any existing connection associated with inbound queued data. A fetched connection context may be retrieved based on lookups associated with the inbound queued data. As TCP processing takes place, dynamic memory access (DMA) operations also take place in parallel to exchange payload data with a host memory. These parallel operations provide at least some of the significant benefits over previous techniques. The figures and text of Exhibit A also discusses other more particular features variously related to, for example, various queues to exchange data with an interface between the TOE and an operating system (OS), a

scheduler of the TOE engine to variously spawn and/or wake up threads, and a core engine operating on the TOE.

6. From at least prior to December 03, 2003 to constructive reduction to practice (application filing on March 31, 2004), due diligence was taken in reducing the invention to practice.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-described application or an patent issued therefrom.

Respectfully submitted,

Date 7th of JULY, 2008 
Sriram R. Vangal